Decomposition of relational schemes

Desirable properties of decompositions
Dependency preserving decompositions
Lossless join decompositions

Desirable properties of decompositions 1

Lossless decompositions

A decomposition of the relation scheme $R$ into subschemes $R_1, R_2, \ldots, R_n$ is **lossless** if, given tuples $r_1, r_2, \ldots, r_n$ in $R_1, R_2, \ldots, R_n$ respectively, such that $r_i$ and $r_j$ agree on all common attributes for all pairs of indices $(i,j)$, the – uniquely defined - tuple derived by joining $r_1, r_2, \ldots, r_n$ is in $R$.

Terminology: "lossless join" decomposition

Desirable properties of decompositions 3

Dependency preserving decompositions

A decomposition of the relation scheme $R$ into subschemes $R_1, R_2, \ldots, R_n$ is **dependency preserving** if all the FDs within $R$ can be derived from those within the relations $R_1, R_2, \ldots, R_n$.

If $F$ is the set of dependencies defined on $R$, then the requirement is that the set $G$ of dependencies that can be obtained as projections of dependencies in $F^+$ onto $R_1, R_2, \ldots, R_n$ together generate $F^+$.

Note carefully that it is not enough to check whether projections of dependencies in $F$ onto $R_1, R_2, \ldots, R_n$ together generate $F^+$.

Desirable properties of decompositions 4

An illustrative example

Replace $SADDRESS$ by $CITY$ and $AGENT$ fields in $SUPPLIERS(SNAME, SADDRESS, ITEM, PRICE)$

Semantics: *Each supplier is based in a city, and the enterprise responsible for setting up the database has an agent for each city.*

Derive in this way a new relation $SCAIP(S, C, A, I, P)$ where $S$ is $SNAME$, $C$ is $CITY$, $A$ is $AGENT$ etc.
Desirable properties of decompositions 5

An illustrative example

SCAIP2

Derive in this way a relation SCAIP(S, C, A, I, P)
where S is SNAME, C is CITY, A is AGENT etc.

The set $F$ of functional dependencies is generated by:

\[ S \rightarrow C, \quad C \rightarrow A, \quad S \rightarrow I \rightarrow P \]

... each supplier sited in one city
... each city has one agent serving it
... each supplier sells each given item at fixed price

Desirable properties of decompositions 6

An illustrative example

SCAIP3

Consider decomposition \{ SCA, SIP \}:

This is lossless: Suppose that the tuples $t_{SCA}$ and $t_{SIP}$ are in the relations SCA and SIP respectively.

If $t_{SCA}$ and $t_{SIP}$ agree on $S$, then their join is a tuple $t_{SCAIP} \equiv (s, c, a, i, p)$, where $c$ and $a$ are determined by the attribute $s$ and the $i$ and $p$ attributes are such that $p$ is determined by $s$ and $i$. Any tuple that satisfies these two FDs is in the relation SCAIP.

Desirable properties of decompositions 7

An illustrative example

SCAIP4

Consider decomposition \{ SCA, SIP \}:

Also dependency preserving: the sets of dependencies
\{ $S \rightarrow C, \quad C \rightarrow A$ \} and \{ $S \rightarrow I \rightarrow P$ \}
are included in the projections of $F^+$ onto SCA and SIP.

This means that the FDs in $F$, from which all dependencies are generated, are explicit in the sub-schemes SCA and SIP in this case.

Desirable properties of decompositions 8

An illustrative example

SCAIP5

In decomposition \{ SIP, SCA \}, have problems with SCA.

E.g. update anomaly if want to store an agent for a city in which no supplier is currently located

Get around this by decomposing SCA further:
- decompose as \{ SC, CA \}
- decompose as \{ SC, SA \}
- decompose as \{ CA, SA \}
Desirable properties of decompositions 9

An illustrative example

\[ F = \{ S \rightarrow C, C \rightarrow A, S \rightarrow P \} \]

- decompose as \{ SC, CA \}
  
  this is both lossless join and dependency preserving

- decompose as \{ SC, SA \}

In this case the images of the FDs in \( F^+ \) on SC and SA are \{S \rightarrow C\} and \{S \rightarrow A\} respectively, but the dependency \( C \rightarrow A \) can't be inferred. So this decomposition is not dependency preserving.

Desirable properties of decompositions 10

An illustrative example

\[ F = \{ S \rightarrow C, C \rightarrow A, S \rightarrow P \} \]

- decompose as \{ CA, SA \}

In this case, have possibility that Fred is agent for Hull and York, and PVC based in Hull. Then:

\[(Hull, Fred) \ast (PVC, Fred) = (PVC, Hull, Fred)\]
\[(York, Fred) \ast (PVC, Fred) = (PVC, York, Fred)\]

The second join is not in the relation SCA. So this decomposition is not lossless join.

Dependency Preserving Decompositions 1

Let \( R \) be a relation scheme, \( \rho \) a decomposition of \( R \) and \( F \) a set of functional dependencies of \( R \).

If \( Z \) is a set of attributes in \( R \), then

\[ \Pi_Z(F) = \{ X \rightarrow Y \in F \mid XY \subseteq Z \} \]

The decomposition \( \rho \) is dependency preserving if \( F \) is logically implied by the union of the sets of functional dependencies \( \Pi_T(F^+) \), where \( T \) ranges over all sub-schemes of \( \rho \).

Dependency Preserving Decompositions 2

Illustrative Example

\( R = ABCD \) and \( \rho = \{AB, BC, CD\} \)
\( F = \{ A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A \} \)

Question: is \( \rho \) dependency preserving?

Certainly \( \{ A \rightarrow B, B \rightarrow C, C \rightarrow D \} \) are captured. How about \( D \rightarrow A \)? Is also, because

\[ F^+ \supseteq \{ B \rightarrow A, C \rightarrow B, D \rightarrow C \} \]

and these FDs are recorded in the sub-schemes \( AB, BC, CD \).

Hence the dependency \( D \rightarrow A \) is also captured.
**Algorithm to check dependency preserving**

OK := true
for each dependency X → Y in F do
begin
Z := X
while changes occur in Z do
    for each sub-scheme T of ρ do
        Z := Z ∪ { A | Z ∩ T → A is in Π_T(F⁺) }
    if not Z ⊇ Y then OK := false
end

**Algorithm to check dependency preserving**

... while changes occur in Z do
    for each sub-scheme T of ρ do
        Z := Z ∪ { A | Z ∩ T → A is in Π_T(F⁺) }
... To compute { A | Z ∩ T → A is in Π_T(F⁺) } calculate
    ((Z ∩ T)+ ∩ T)
where the closure (Z ∩ T)+ is computed with respect to F over the entire relation scheme R.

This avoids need to compute F⁺.

**Illustrating the algorithm in action**

Consider the relation scheme R = ABCD, the dependencies F = { A → B, B → C, C → D, D → A }, and the decomposition ρ = { AB, BC, CD }

Clear that A → B, B → C and C → D are preserved
... can prove that the dependency D → A is preserved by applying the algorithm

Computation of {D}⁺ over R using F yields {A,B,C,D}

Computation of {D}⁺ over R using F yields {A,B,C,D}

Z={D} initially. At each iteration of the while-loop, the algorithm introduces a new attribute into Z. For instance, on the first pass, introduce C when T = CD, on second pass, then introduce B when T = BC etc. Hence:
Z₀ = {D}, Z₁ = {C,D}, Z₂ = {B,C,D}, Z₃ = {A,B,C,D}
where Zᵢ is the value of Z after the iᵗʰ iteration.

This proves that dependency D → A is preserved.
Lossless join decomposition

Let R be a relation scheme, ρ a decomposition of R and F a set of functional dependencies of R. Suppose that the sub-schemes in ρ are \{R_1, R_2, ..., R_k\}.

ρ has lossless join if every extensional part r for R that satisfies F is such that r = \(\Pi_1(r) \times \Pi_2(r) \times ... \times \Pi_k(r)\), where \(\Pi_i(r)\) denotes the projection of r onto \(R_i\).

Informally: r is the natural join of its projections onto the sub-schemes \(R_1, R_2, ..., R_k\).

Examples (revisited as a reminder)

\[
SCAIP = SIP |\times| SCA = SIP |\times| SC |\times| CA \quad \text{lossless}
\]
\[
SCA \subseteq SA |\times| CA \quad \text{and} \quad SCA \neq SA |\times| CA \quad \text{lossy}
\]

... have possibility that Fred is agent for Hull and York, and that PVC is a supplier based in Hull. Then:

\[
(Hull, Fred) \star (PVC, Fred) = (PVC, Hull, Fred)
\]
\[
(York, Fred) \star (PVC, Fred) = (PVC, York, Fred)
\]

The second join is not in the relation SCA.
So this decomposition is not lossless join.

Principles of lossless join decomposition

Let \(\rho = \{R_1, R_2, ..., R_k\}\) be a decomposition of R.

Define the mapping \(m_\rho(\cdot)\) on possible extensions for the relation scheme R [whether or not they satisfy the functional dependencies in R, if there are any], via:

\[
m_\rho(r) = \Pi_1(r) \times \Pi_2(r) \times ... \times \Pi_k(r),\]

where \(\Pi_i(r)\) denotes the projection of r onto sub-scheme \(R_i\).

Notation: use \(r_i\) to denote \(\Pi_i(r)\), for \(1 \leq i \leq k\).

Principles of lossless join decomposition (cont.)

Lemma: With R, ρ and \(r_i\) as above

a) \(r \subseteq m_\rho(r)\)

b) if \(s = m_\rho(r)\), then \(\Pi_i(s) = r_i\)

c) \(m_\rho(m_\rho(r)) = m_\rho(r)\)

The condition on \(m_\rho(\cdot)\) specified in part c) identifies it as a closure operation.

Cf. closure of an interval of real numbers e.g. \(1 < \alpha \leq 2\)
Proof of lemma

a) let \( t \in r \). Then \( \Pi_i(t) \in r_i \) showing that
\[
\Pi_i(t) \mid \Pi_2(t) \mid \ldots \mid \Pi_k(t) = \rho_i(r)
\]
b) by part a) \( r \subseteq \rho_i(r) = s \), so that \( \Pi_i(s) \supseteq r_i \).
But if \( t \in s \), then projection of \( t \) onto sub-scheme \( R_i \) is in \( r_i \) by definition of natural join, so that \( \Pi_i(s) \subseteq r_i \) also.

c) \( \rho_i(\Pi_i(r)) = \Pi_i(s) \) by definition of \( s \)
\[
= \Pi_i(s) \mid \Pi_2(s) \mid \ldots \mid \Pi_k(s)
\]
\[
= \Pi_i(r) \mid \Pi_2(r) \mid \ldots \mid \Pi_k(r)
\]
\[
= \rho_i(r) \) using definition of \( \rho_i \) and part b).

Testing for lossless join decomposition
assuming all data dependencies in \( R \) to be functional

Input: A relation scheme \( R = A_1 A_2 \ldots A_n \), a set of functional dependencies \( F \), and a decomposition
\[
\rho = \{ R_1, R_2, \ldots, R_k \}
\]

Output: \( \rho \) is or is not a lossless join decomposition

Construct table of \( \alpha \)'s and \( \beta \)'s, and repeatedly transform the rows by taking account of the FDs until either one row is all \( \alpha \)'s or no further transformation is possible ...

Principle of algorithm: devise a symbolic representation for tuples \( s_1, s_2, \ldots, s_k \) from \( R_1, R_2, \ldots, R_k \) respectively that are joinable, and for tuples \( t_1, t_2, \ldots, t_k \) in \( R \) so that \( s_i \) is projection of \( t_i \) onto \( R_i \) for each \( i \). Impose all those conditions on \( t_1, t_2, \ldots, t_k \) that follow from the FDs in \( F \). If none of the \( t_i \)'s is the join of \( s_1, s_2, \ldots, s_k \), then they define an extension for \( R \) that exhibits a lossy join.
Lossless Join Decompositions 9

Method of testing for lossless join decomposition

1. Construct a table
   with n columns (corresponding to attributes)
   with k rows (corresponding to sub-schemes)

   Initialise the table at row i column j
   by entering $\alpha_j$ if attribute $A_j$ appears in $R_i$
   and by entering $\beta_{ij}$ otherwise

   NB $\alpha$'s represent joinable tuples, padded out to R by $\beta$'s

Lossless Join Decompositions 10

Method of testing for lossless join decomposition (cont.)

2. Repeatedly modify the table to take account of all dependencies until no further updates occur
   i.e. if $X \rightarrow Y$ and two rows agree on all the attributes in X then modify them so that they also agree on all attributes in Y. Explicitly, change attributes in Y thus:
   - if one symbol is an $\alpha_i$ make the other an $\alpha_i$
   - if both symbols are of form $\beta_{ij}$ make both $\beta_{ij}$ or $\beta_{ij}'$ arbitrarily.

   On termination declare lossless join if and only if one of the rows is $\alpha_1\alpha_2 \ldots \alpha_n$.

Lossless Join Decompositions 11

Illustrative example

Verify the decomposition $\text{SCAI P} = \text{SIP} \times \text{SC} \times \text{CA}$

is a lossless join ....

Initial table

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>C</th>
<th>A</th>
<th>I</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP</td>
<td>$\alpha_1$</td>
<td>$\beta_{12}$</td>
<td>$\beta_{13}$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
</tr>
<tr>
<td>SC</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\beta_{23}$</td>
<td>$\beta_{24}$</td>
<td>$\beta_{25}$</td>
</tr>
<tr>
<td>CA</td>
<td>$\beta_{31}$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\beta_{34}$</td>
<td>$\beta_{35}$</td>
</tr>
</tbody>
</table>

Functional dependencies are $S \rightarrow C$, $C \rightarrow A$, $S I \rightarrow P$

Lossless Join Decompositions 12

Illustrative example

Functional dependencies are $S \rightarrow C$, $C \rightarrow A$, $S I \rightarrow P$

and from these arrive via stage 2 of algorithm at table:

<table>
<thead>
<tr>
<th></th>
<th>S</th>
<th>C</th>
<th>A</th>
<th>I</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
</tr>
<tr>
<td>SC</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\beta_{24}$</td>
<td>$\beta_{25}$</td>
</tr>
<tr>
<td>CA</td>
<td>$\beta_{31}$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\beta_{34}$</td>
<td>$\beta_{35}$</td>
</tr>
</tbody>
</table>

at which point no further dependencies apply.

Row 1 shows that the result is lossless
**Lossless Join Decompositions 13**

Principle of the lossless join algorithm illustrated ...

Consider the example: in the initial table

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
<th>A</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP</td>
<td>α₁</td>
<td>β₁₂</td>
<td>β₁₃</td>
</tr>
<tr>
<td>SC</td>
<td>α₁</td>
<td>α₂</td>
<td>β₂₃</td>
</tr>
<tr>
<td>CA</td>
<td>β₃₁</td>
<td>α₂</td>
<td>β₃₄</td>
</tr>
</tbody>
</table>

... the rows can be seen as representing generic tuples from SIP, SC and CA that are joinable (i.e. agree on all common attributes). The join of these three tuples will necessarily be $α₁α₂α₃α₄α₅$.

**Lossless Join Decompositions 14**

Key question: are the functional dependencies enough to ensure that $α₁α₂α₃α₄α₅$ is itself a tuple in the relation SCAIP?  
After modification to take account of all FDs, suitable tuples matching the template for equality of values in the 3 rows in the table define a valid extensional part for SCAIP: can substitute them to get a concrete relation $r$.

Either one of the 3 tuples is $α₁α₂α₃α₄α₅$ lossless or $α₁α₂α₃α₄α₅ \in m_ρ(r) \setminus r$ lossy.

**Lossless Join Decompositions 15**

Algorithm shows that SCA is a lossy join of SA and CA:

<table>
<thead>
<tr>
<th>FDs are $S \rightarrow C$, $C \rightarrow A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial and final form of table</td>
</tr>
<tr>
<td>SA $α₁$ $β₁₂$ $α₃$</td>
</tr>
<tr>
<td>CA $β₂₁$ $α₂$ $α₃$</td>
</tr>
</tbody>
</table>

Fred is agent for Hull [$β₁₂$] and York, PVC is based in Hull, there is another supplier [$β₂₁$] say GPT at York.

Take as extension of SCA the pair of valid tuples:

(PVC, Hull, Fred) [row 1] and (GPT, York, Fred) [row 2]

Project onto SA and CA, get

(PVC, Fred), (Hull, Fred), (GPT, Fred), (York, Fred)

Take natural join to get rogue tuples:

(PVC, York, Fred) [$α₁$, $α₂$, $α₃$], (GPT, Hull, Fred) [$β₂₁$, $β₁₂$, $α₃$]

**Lossless Join Decompositions 16**

Theorem

If $ρ = \{S, T\}$ is a decomposition of $R$, and $F$ is the set of FDs for $R$, then $ρ$ is a lossless join decomposition with respect to $F$ if and only if

$$\text{either } T \setminus S \subseteq (S \cap T)^+ \text{ or } S \setminus T \subseteq (S \cap T)^+. $$

Proof: Applying the method of the algorithm to test for lossless join, get initial table of the form:

<table>
<thead>
<tr>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$αₐₐ$</td>
<td>$βββ$</td>
</tr>
<tr>
<td>$S$</td>
<td>$aₐₐ$</td>
</tr>
</tbody>
</table>
Theorem

If $\rho = \{S, T\}$ is a decomposition of $R$, and $F$ is the set of FDs for $R$, then $\rho$ is a lossless join decomposition with respect to $F$ if and only if either $S \setminus T \subseteq (S \cap T)^+$ or $S \setminus T \subseteq (S \cap T)^+$.

Proof (cont.) ... get initial table of the form:

<table>
<thead>
<tr>
<th>$S \cap T$</th>
<th>$S \setminus T$</th>
<th>$T \cap S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$\alpha \ldots \alpha$</td>
<td>$\beta \ldots \beta$</td>
</tr>
<tr>
<td>$S$</td>
<td>$\alpha \ldots \alpha$</td>
<td>$\alpha \ldots \alpha$</td>
</tr>
</tbody>
</table>

The final table is this table modified so that every column labelled by an attribute in $(S \cap T)^+$ is changed to an $\alpha$, from which the theorem follows.

Application of Thm: SCA is a lossy join of SA and CA, as neither of the dependencies $A \rightarrow S$, $A \rightarrow C$ is valid.

Corollary to the theorem: If $R$ is a relation scheme, and $X \rightarrow A$ is a functional dependency in $R$, where $A$ is an attribute, $X$ is a set of attributes not containing $A$, and $XA$ is a proper subset of $R$, then $R_1=XA$, $R_2=R\setminus A$ is a lossless join decomposition of $R$.

Proof: $R_1 \cap R_2 \supseteq X$, hence $R_1 \setminus R_2 = A \in (R_1 \cap R_2)^+$.

Exercise for lossless join algorithm from Ullman 1982:

Take $R = ABCDE$

$R_1 = AD$, $R_2 = AB$, $R_3 = BE$, $R_4 = CDE$, $R_5 = AE$

with the functional dependencies

$A \rightarrow C$, $B \rightarrow C$, $C \rightarrow D$, $DE \rightarrow C$, $CE \rightarrow A$

In this example, the identification of $\beta_j$’s is crucial.

Can trace the algorithm through three stages ...
Lossless Join Decompositions 21

Split $R = ABCDE$ into $R_1=AD$, $R_2=AB$, $R_3=BE$, $R_4=CDE$, $R_5=AE$ with the FDs $A \rightarrow C$, $B \rightarrow C$, $C \rightarrow D$, $DE \rightarrow C$, $CE \rightarrow A$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>$\alpha_1$</td>
<td>$\beta_{12}$</td>
<td>$\beta_{13}$</td>
<td>$\alpha_4$</td>
<td>$\beta_{15}$</td>
</tr>
<tr>
<td>AB</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\beta_{13}$</td>
<td>$\beta_{24}$</td>
<td>$\beta_{25}$</td>
</tr>
<tr>
<td>BE</td>
<td>$\beta_{31}$</td>
<td>$\alpha_2$</td>
<td>$\beta_{13}$</td>
<td>$\beta_{34}$</td>
<td>$\alpha_5$</td>
</tr>
<tr>
<td>CDE</td>
<td>$\beta_{41}$</td>
<td>$\beta_{42}$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
</tr>
<tr>
<td>AE</td>
<td>$\alpha_1$</td>
<td>$\beta_{52}$</td>
<td>$\beta_{13}$</td>
<td>$\beta_{54}$</td>
<td>$\alpha_5$</td>
</tr>
</tbody>
</table>

Lossless Join Decompositions 22

Split $R = ABCDE$ into $R_1=AD$, $R_2=AB$, $R_3=BE$, $R_4=CDE$, $R_5=AE$ with the FDs $A \rightarrow C$, $B \rightarrow C$, $C \rightarrow D$, $DE \rightarrow C$, $CE \rightarrow A$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>$\alpha_1$</td>
<td>$\beta_{12}$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\beta_{15}$</td>
</tr>
<tr>
<td>AB</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\beta_{25}$</td>
</tr>
<tr>
<td>BE</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
</tr>
<tr>
<td>CDE</td>
<td>$\alpha_1$</td>
<td>$\beta_{42}$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
</tr>
<tr>
<td>AE</td>
<td>$\alpha_1$</td>
<td>$\beta_{52}$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
</tr>
</tbody>
</table>